

Using Non-invertible Data Transformations to Build Adversarial-Robust Neural Networks

Qinglong Wang^{1,2}, Wenbo Guo^{1,3}, Alexander G. Ororbia II¹, Xinyu Xing¹, Lin Lin¹, C. Lee Giles¹,
Xue Liu², Peng Liu¹ and Gang Xiong⁴

¹Pennsylvania State University

²McGill University

³Shanghai Jiao Tong University

⁴Chinese Academy of Sciences

Abstract—Deep neural networks have proven to be quite effective in a wide variety of machine learning tasks, ranging from improved speech recognition systems to advancing the development of autonomous vehicles. However, despite their superior performance in many applications, these models have been recently shown to be susceptible to a particular type of attack possible through the generation of particular synthetic examples referred to as adversarial samples. These samples are constructed by manipulating real examples from the training data distribution in order to “fool” the original neural model, resulting in misclassification (with high confidence) of previously correctly classified samples. Addressing this weakness is of utmost importance if deep neural architectures are to be applied to critical applications, such as those in the domain of cybersecurity. In this paper, we present an analysis of this fundamental flaw lurking in all neural architectures to uncover limitations of previously proposed defense mechanisms. More importantly, we present a unifying framework for protecting deep neural models using a non-invertible data transformation—developing two adversary-resilient architectures utilizing both linear and nonlinear dimensionality reduction. Empirical results indicate that our framework provides better robustness compared to state-of-art solutions while having negligible degradation in accuracy.

I. INTRODUCTION

Aside from its highly publicized victories in Go [33], there have been numerous successful applications of deep neural network (DNN) learning in image and speech recognition. Recent interest has been to integrate it into critical fields like medical imaging [1], [37] and self-driving cars [13], [8]. In cybersecurity, security companies have demonstrated that deep learning could offer a far better way to detect all types of malware [27], [4], [38] and recognize such functions in binary code [32].

However, recent work[35], [22] uncovered a potential flaw in DNN-powered systems that could be readily exploited by

an attack. They show that an attacker could use the same learning algorithm, back-propagation, and a surrogate data set to construct an auxiliary model that can accurately approximate the target DNN model. When compared to the original model that usually returns categorical classification results (e.g., benign or malicious binary code), such an auxiliary model could provide the attacker with useful details about a DNN’s weaknesses (e.g., a continuous classification score for a malware sample). As such, the attacker can use the auxiliary model to perform invertible computation, or examine class/feature importance to identify the features that have significant impact on the target’s classification ability. With this knowledge of feature importance, the attacker can minimize effort in crafting an *adversarial sample* – a synthetic example generated by modifying a real example slightly in order to make the original DNN model believe it belongs to the wrong class with high confidence. For example, Fig. 1 illustrates the case where an adversarial sample constructed by modifying a picture of a panda misleads the original DNN model into believing it is a gibbon.

To mitigate the aforementioned kind of attack, solutions [11], [15], [20] proposed generally follow the basic idea of *adversarial training* in which a DNN is trained with both samples from the original data distribution as well as artificially synthesized adversarial ones. A recent unification of previous approaches [24] showed that they were all special cases of a general, regularized objective function, *DataGrad*. While such a framework vastly improves the DNN’s robustness to adversarial samples, the final model is still not completely resilient given that the adversarial sample space is unbounded. Hence, a newly trained DNN would only be robust with respect to previously observed adversarial samples (or for those relatively near to training samples of the underlying manifold if one uses the general approximation to *DataGrad* developed in [24]).

Here, we present a new defense framework that increases the difficulty for attackers by crafting adversarial samples, i.e. making DNN models resilient to any adversarial sample whether or not they have been previously observed by DNN models. The basic idea underlying our new framework is to transform the input data into a new representation before presenting it to a DNN model. More specifically, the transformation employs a non-invertible dimensionality reduction approach that projects an input sample into a low dimension

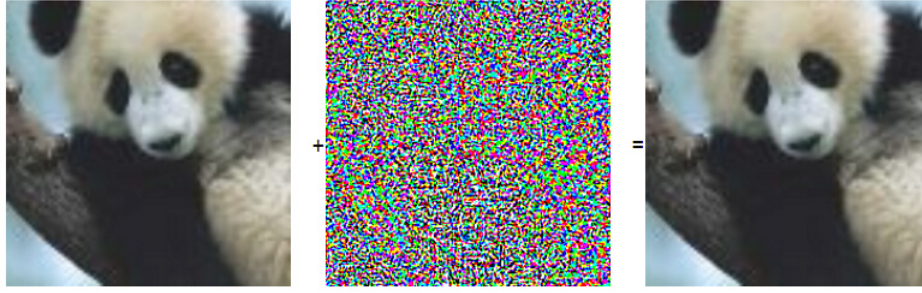


Figure 1. Demonstration of an adversarial example generated from a panda picture [10]. The left picture represents the original image classified as a panda with 60% confidence. Right picture is an adversarial sample obtained by adding a tiny perturbation to the original picture. Despite being visually similar to the original image, the adversarial sample is classified as a gibbon with 99% confidence.

representation. As mentioned earlier, an attacker needs to perform an invertible computation to examine the feature/class importance through an auxiliary DNN model. With this transformation, the complexity of such an invertible computation is significantly increased and the attacker is no longer able to perform feature/class examination.

Technically speaking, our framework is similar to a distillation process, which trains a DNN model using the knowledge transferred from a different model [26]. However, it is fundamentally different from the distillation approach. In fact, the distillation approach does not improve the robustness of a DNN because it does not increase the complexity of the invertible computation. As we will demonstrate in Section III, an attacker can easily construct an auxiliary model and craft adversarial samples even when the DNN model is trained through a distillation procedure.

In summary, this work makes the following contributions.

- We conduct a detailed analysis on various existing defense mechanisms against adversarial samples, and demonstrate their limitations.
- We present a comprehensive framework that makes a DNN model resilient to adversarial samples by integrating an input transformation into the DNN model.
- We develop two new defense mechanisms by injecting different dimensional reduction methods into the proposed framework.
- We theoretically and empirically evaluate the DNN models, showing that our new defense framework is resilient to adversarial samples.

The rest of this paper is organized as follows. Section II introduces the background of DNNs. Section III discusses the limitations of existing defense mechanisms. Section IV presents a unifying framework and Section V develops two defense mechanisms using dimensionality reduction methods, both linear and nonlinear. In Section VI, we evaluate our proposed framework. Section VII discusses some related work followed by the conclusion in Section VIII.

II. BACKGROUND

We briefly introduce the well-established deep neural network (DNN) model and then describe how to generate adver-

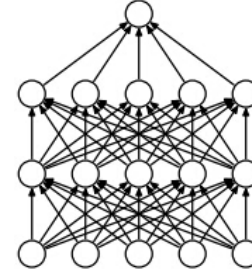


Figure 2. Standard neural network [34] with two hidden layers, where neurons are fully connected. This architecture is often referred to as a feed-forward neural network [2]

sarial samples from it in order to exploit its inherent flaws. Finally, we discuss our threat model.

A. Deep Neural Networks

A typical DNN architecture consists of multiple successive layers of processing elements, or so-called “neurons”. Each processing layer can be viewed as learning a different, more abstract representation of the original multidimensional input distribution. As a whole, a DNN can be viewed as a highly complex function that is capable of mapping original high-dimensional data points to a lower dimensional space in any nonlinear fashion. As shown in Fig 2, a typical DNN contains an input layer, multiple hidden layers, and an output layer. The input layer takes in each data sample in the form of a multidimensional vector. Starting from the input, computing the activations of each subsequent layer simply requires, at minimum, a matrix multiplication (where a weight/parameter vector, with length equal to the number of hidden units in the target layer, is assigned to each unit of the layer below) followed by summation with a bias vector. This process roughly models the process of a layer of neurons integrating the information received from the layer below (i.e., computing a pre-activation) before applying an elementwise activation function¹. This integrate-then-fire process is repeatedly subsequently for each layer until the last layer is reached. The last layer, or output, is generally interpreted as the model’s predictions for some given input data, and is often designed compute a parameterized

¹There are many types of activations to choose from, including the hyperbolic tangent, the logistic sigmoid, or the linear rectified function, etc [2]

posterior distribution using the softmax function (also known as multi-class regression or maximum entropy). This bottom-up propagation of information is also referred to as *feed-forward* inference [14].

During the learning phase of the model, the DNN's predictions are evaluated by comparing them with known target labels associated with the training samples. Specifically, both predictions and labels are taken as the input to a selected cost function, such as cross-entropy. The DNN's parameters are then optimized with respect to the cost function using the method of steepest gradient descent, minimizing prediction errors on the training set. Parameter gradients are calculated using back-propagation of errors [31]. Since the gradients of the weights represent their influence on the final cost, there have been multiple algorithms developed for finding optimal weights more accurately and efficiently [21].

B. The Adversarial Sample Problem

Even though a well trained model is capable of recognizing out-of-sample patterns, a deep neural architecture can be easily fooled by introducing perturbations to the input variables that are often indistinguishable to the human eye [35]. These so-called “blind spots”, or adversarial samples, exist because the input space of DNN is unbounded [10]. Based on this fundamental flaw, we can uncover specific data samples in the input space able to bypass DNN models. More specifically, it was studied in [10] that attackers can find the most powerful blind spots via effective optimization procedures. In multi-class classification tasks, such adversarial samples can cause a DNN model to classify a data point into a random class besides the correct one (sometimes not even a reasonable alternative).

Furthermore, according to [35], DNN models that share the same design goal, for example recognizing the same image set, all approximate a common highly complex, nonlinear function. Therefore, a relatively large fraction of adversarial examples generated from one trained DNN will be misclassified by the other DNN models trained from the same data set but with different hyper-parameters. Therefore, given a target DNN, we refer to adversarial samples that are generated from other different DNN models but still maintain their attack efficacy against the target as *cross-model adversarial samples*.

Adversarial samples are generated by computing the derivative of the cost function with respect to the network's input variables. The gradient of any input sample represents a direction vector in the high dimensional input space. Along this direction, an small change of this input sample will cause a DNN to generate a completely different prediction result. This particular direction is important since it represents the most effective way of compromising the optimization of the cost function. Discovering this particular direction is realized by passing the layer gradients from output layer to input layer via back-propagation. The gradient at the input may then be applied to the input sample(s) to craft an adversarial example(s).

To be more specific, let us define a cost function $\mathcal{L}(\theta, X, Y)$, where θ , X and Y denotes the weights of the DNN, the input data set, and the corresponding labels respectively. In general, adversarial samples can be crafted by adding to legitimate ones via an *adversarial perturbation* δx . In [10],

the efficient and straightforward *fast gradient sign* method was proposed for calculating adversarial perturbation as shown in in (1):

$$\delta x = \phi \text{sign}(\mathcal{J}_{\mathcal{L}}(\theta, x, y)), \quad (1)$$

here δx is calculated by multiplying the sign of the gradients of legitimate samples with some coefficient ϕ . $\mathcal{J}_{\mathcal{L}}$ denotes the gradients of the loss function $\mathcal{L}(\cdot)$ with respect to the data x . y is the corresponding label of x . ϕ controls the scale of the gradients to be added.

Adversarial perturbation indicates the actual direction vector to be added to legitimate samples. This vector drives a legitimate sample x towards a direction that the cost function $\mathcal{L}(\cdot)$ is significantly sensitive to. However, it should be noted that δx must be maintained within a small scale. Otherwise adding δx will cause a significant distortion to a legitimate sample, leaving the manipulation to be easily detected. In the next section, we will develop two threat models that utilize the properties of adversarial samples.

C. Threat Model

The threat models introduced in this part all follow the line of exploiting the sensitivity of DNN models with respect to input samples. We first introduce an attack which utilizes direct knowledge of the target DNN model. This attack is only valid under the assumption that all detailed DNN information, regarding both architecture and exact parameters, is disclosed to an adversary. Since this assumption might be too strong to be practical in real world scenarios, we further present another threat model that is also effective for fooling a normal DNN but not restricted by the aforementioned assumption.

1) *White Box Threat Model*: As previously introduced, an adversary can manipulate data samples to mislead a normal DNN by using gradients with respect to a loss function. In order to obtain the gradient of a data sample, the adversary needs to have access to the exact form of the cost function, the architecture of the DNN mode, as well as the tuned coefficients found as a result of the model's training procedure. Assuming that all of this information is available, an adversary can tweak any testing data sample towards a direction using its gradient information.

To be more specific, the gradient of a test sample indicates the direction along which even small changes of this test sample will yield a significant difference in the cost function. Once the gradients are calculated, one may use the procedure detailed in Section II-B to manipulate samples to trick the DNN into generating worst case prediction results. However, in the real world, detailed DNN information can be well protected using various data integrity techniques making the White Box assumption too strong. Therefore, we introduce another threat model not restricted by this assumption and is yet effective in attacking a normal neural architecture.

2) *Black Box Threat Model*: Since DNNs have become a well established machine learning method, the training procedures and typical cost functions used are common public knowledge. Furthermore, recall that a DNN is designed to approximate a highly nonlinear function that is capable of mapping original data samples into a space that makes prediction simpler. If we assume that a given training data and

application scenario, then it follows that different DNN models are all tuned to behave similarly to one other. Therefore, an attacker could utilize a well-known learning algorithm, a typical architecture, and a similar data set to build an auxiliary model that accurately approximates the target model. Since the auxiliary model is in the adversary’s full possession, he can easily generate cross-model adversarial samples as previously introduced. Therefore, a successful attack (using cross-model adversarial samples) can be conducted even most of the information about a target DNN is unavailable.

Based on these two threat models, our proposed DNN architecture is designed to be robust in both scenarios. Particularly, we provide both a theoretical proof and empirical demonstration of the robustness of our proposed DNN architecture even under the white box threat. In the following section, we will provide a thorough analysis of existing solutions for defending against adversarial samples. We will then show that the fundamental flaw introduced in this section is not fully and properly addressed by these solutions.

III. MOTIVATION

We start this section with an overview of recent solutions developed for defending against adversarial samples. These solutions fall into two categories: 1) augmenting the training set and 2) enhancing model complexity. The former is mainly represented by adversarial training while the latter mainly combines various data transformations with a standard DNN.

A. Data Augmentation

For image recognition, current publicly available image sets can be easily used as a starting point. Adversarial samples shown in Fig. 1 are blind spots specifically created to trick a DNN into misclassifying with high confidence. To resolve the blind spot issue, there have been many data augmentation methods proposed for deep learning tasks [10], [24]. In principle, these methods expand their training set by combining known samples with potential blind spots. The same mechanism has also been employed for defending against adversarial samples, also known as adversarial training [24]. Here, we analyze the limitations of data augmentation mechanisms and argue that these limitations also apply to adversarial training methods.

Given the high dimensionality of data distributions that DNNs typically learn from, the input space is generally considered infinite [10]. This implies that there could also exist an infinite amount of blind spots, which are adversarial samples specific to DNN models. Therefore, data augmentation based approaches have the challenge of covering these very large spaces. Since adversarial training is a form of data augmentation, this approach is inherently limited in its ability to protect a DNN from blind spots.

Adversarial training can be formally described as adding a regularization term known as *DataGrad* to a DNN’s training cost function [24]. The regularization penalizes the directions uncovered by adversarial perturbations (introduced in Section II). Therefore, adversarial training works to improve the worst case performance of DNN. Treating the DNN much like a generative model, adversarial samples are produced via

back-propagation and mixed into the training set and directly integrated into the model’s learning phase.

Despite the fact that there exists infinite adversarial samples, adversarial training is effective for defending against those which are powerful and easily crafted. This is due to the fact that, in most adversarial training approaches [24], [10], adversarial samples can be generated efficiently for a particular type of DNN. The fast gradient sign method [10] can generate a large pool of adversarial samples quickly while *DataGrad* [24] focuses on dynamically generating them per every parameter update. However, the simplicity and efficiency of generating adversarial samples also makes adversarial training vulnerable when these two properties are exploited in order to attack the adversarial training *itself*. Given that there exist infinite adversarial samples, we would need to repeat an adversarial training procedure each time a new adversarial example is found. *DataGrad*, as mentioned [24], could be viewed as taking advantage of adversarial perturbations to better explore the underlying data manifold—however, while this leads to improved generalization, it does not offer any guarantees in covering all possible blind-spots. More importantly, adversarial training does nothing to change the actual reversible nature of a DNN’s architecture itself, which we argue is the more direct and ultimately more effective way in building a strong defense.

B. Enhancing Model Complexity

DNN models are already complex, with respect to both the nonlinear function they try to approximate as well as their layered composition of many parameters. The architecture is straightforward in order to facilitate the flow of information forwards and backward, which greatly alleviates the effort in generating adversarial samples. Therefore, several ideas [26], [11] have been proposed to enhance the complexity of DNN architecture, aiming to improve the tolerance of complex DNN models with respect to adversarial samples generated from simple DNN models.

[26] develops a *defensive distillation* mechanism, which trains a DNN from data samples that are distilled by another DNN. By using the knowledge transferred from the other DNN, the learned DNN classifiers become less sensitive to adversarial samples. However, although shown to be effective, this method still relies on gradient flow from output to input. This is because both DNN models used in this architecture can be approximated by an adversary via training two other DNN models that share the same functionality and have similar performance. Once the two approximating DNN models are obtained, the adversary can generate adversarial samples specific to this distillation-enhanced DNN model. [11] proposed stacking an auto-encoder together with a normal DNN, similar to [26]. It was first shown that this auto-encoding enhancement increases DNN resilience to adversarial samples. However, the authors further demonstrate that this stacked model can be simply taken as a new DNN and easily generate new adversarial samples.

Though the above approaches, both from data-augmentation and model complexity perspectives, have proven effective in handling samples generated from normal adversarial DNN models, they do not handle all adversarial samples. In light of this, we propose a framework that blocks

the gradient flow from the output to input variables, a solution that prove effective even when the architecture and parameters of a given a DNN are publicly disclosed.

IV. DATA TRANSFORMATION ENHANCED DNN FRAMEWORK

In this section, we shall fully specify our framework’s design goals and choose a particular type of data transformation that will fulfill these goals.

A. Design Goals

Recall that many previously proposed solutions, especially adversarial training methods [10], [24], can be classified as forms of data augmentation. By their very nature, these approaches cannot possibly hope to cover the entire data space, unless perhaps they had access to all important representative points of the underlying manifold (which is highly unlikely in practice). This implies that attackers can always find adversarial samples targeted to any particular DNN model. More importantly, developing adversarial training methods has required the invention of efficient methods for generating adversarial samples [10], [35], consequently providing more useful tools for attackers.

This further facilitates new attacks specifically designed for DNN models learned using adversarial training algorithms.

We argue that a robust DNN architecture has the property that adversarial samples cannot be generated from itself. This we consider to mean that the flow of error gradients from the output to input variables must be obstructed (while minimally affecting model generalization performance). Since most DNN models are simple and straightforward, their architectures are usually well known and hence easy to approximate. Therefore, a robust DNN satisfying our desired property can still be secure even under the most extreme circumstances, such as when all of its detailed parameters are disclosed to adversaries. Our framework specifically must satisfy two goals:

- It has minimal impact on the performance of a DNN model when legitimate samples are seen.
- It is resilient to cross-model adversarial samples as introduced in Section II.
- It ‘blocks’ gradient flow from reaching the input.

B. Framework Overview

Adversarial samples are generated via backpropagation during training, as described in Section II. This implies that by following the path of gradient flow, adversarial samples can be generated quite conveniently. In order to prevent this, we propose blocking this gradient path by combining the DNN architecture with particular data transformation methods. Specifically, we insert a data transformation module before the input layer of the DNN, in effect performing a special preprocessing of input samples. As a result, the procedure for generating adversarial samples will only work for the transformed data, if the right kind input transformation is chosen such that information cannot reach the original observed variables directly from objective function. Furthermore, even when an adversary successfully generates adversarial samples

based on the transformed data, we can *guarantee* that this adversary will be unable to map the adversarial data samples to the original input space if the data transformation employed is *non-invertible*. More formally, we define a data transformation method as non-invertible when it satisfies either of the following properties: (1) inverting the data transformation is computationally too complex to be tractable; or (2) inverting the data transformation will cause significant reconstruction error.

Our proposed framework, graphically depicted in Fig 3, has several properties that guarantee that a DNN is well protected against adversarial samples while suffering at most only trivial changes in performance. First, any non-invertible data transformation method can ensure that the gradient flow in the framework is effectively blocked. This critically ensures that an adversary cannot generate samples using our framework using a method like the fast gradient sign method (see Section II). Second, the framework is capable of legitimately handling test samples since the DNN model is trained on top of a data transformation that preserves crucial information contained in the original input distribution. Third, any non-invertible data transformation method may be used in tandem with a normal DNN model under our unified framework.

However, it is important to note that integrating a DNN with a data transformation may be computationally expensive, which might make this defense less attractive given that standard DNN training is already computationally intensive. We resolve this issue by exploring data transforms that are computationally efficient and more importantly, incremental. The latter requirement is essential given that any data transformation method must be capable of handling unseen samples as they are presented. Otherwise, the data transformation will need to be retrained, and subsequently, the DNN on top of the newly retrained transform layer.

Dimensionality reduction is one particular data transformation mechanism that satisfies these design objectives. First, dimensionality reduction methods are often designed to preserve at least the most important aspects of the original data. Second, dimensionality reduction can serve as a filter for adversarial perturbations when a DNN is confronted with cross-model adversarial samples. Third, dimensionality reduction helps reduce the dimensionality of the input distribution that is fed into the DNN. Finally, it is easier to develop non-invertible data transformation methods, since recovering higher dimensional data from lower dimensional data is difficult. The following sections introduce details of two developed defence mechanisms using different non-invertible dimensional reduction methods.

V. DATA TRANSFORMATION ENHANCED DNNs

We present two variations of an adversary resilient DNN architecture that satisfies the two key design goals of our proposed defense framework (Section IV). In particular, one variant utilizes a linear method while the other makes use of a non-linear approach. With respect to the linear method, we show that, theoretically, there is a lower bound on the reconstruction error. More importantly, we design our linear method to have a reconstruction error that is significantly larger than this lower bound which in turn satisfies the characteristics we defined for a non-invertible data transformation.

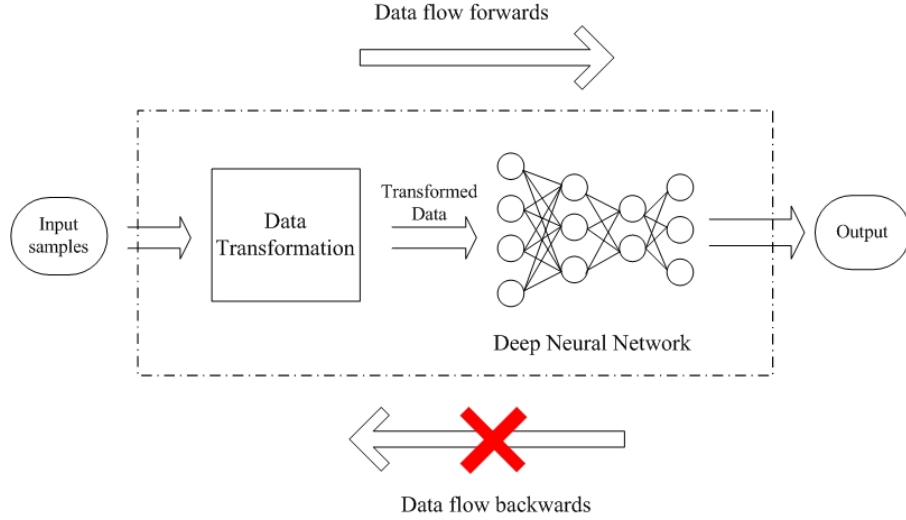


Figure 3. A data transformation enhanced DNN. In the feed-forward pass, input data is first projected to a particular space and the transformed data is then used to train a standard DNN classifier. However, the gradient flow is blocked by the data transformation.

With respect to the second approach, we employ a nonlinear dimension reduction approach that we prove satisfies the second characteristic of a non-invertible data transformation. Critically, we show that, in order to recover the original data from this low dimensional representation, an inversion of the corresponding dimensional reduction method is required, which can be converted to a quadratic problem with a non-positive, semi-definite constraint. This belongs to a class of NP-hard problems, according to [5].

A. Designed Linear Mapping (DLM) DNN

We first propose a novel linear dimensional reduction method, which stems from principal component analysis (PCA). Then, a straightforward strategy is introduced for reconstructing the original high dimensional data by inverting our proposed transformation. In addition, we provide a theorem that places the lower bound on the reconstruction error. The inversion strategy will be further used in Section VI to demonstrate that our proposed method satisfies the requirements of non-invertibility.

PCA is one of the most widely adopted dimensional reduction methods, especially since it is computationally efficient and easy to implement [16]. These are useful properties when considering PCA as a possible transform to combine with a DNN, since the DNN itself is already computationally expensive. Additionally, PCA preserves critical information by finding a low-dimensional subspace with maximal variance. However, PCA can be easily inverted to reconstruct original data samples given their lower dimensional mappings. Moreover, the reconstructed data usually exhibits insignificant distortion when compared to the original data (see Figure 7 (a)). This implies that, if an attacker generates adversarial samples from a DNN trained on low dimensional mappings, it would be simple to generate high dimension adversarial samples by mapping this low dimensional data back to the original dimension. We will show in Section VI the effectiveness of inverting PCA by examining several reconstructed images from the MNIST data set.

PCA preserves meaningful features of the original data when mapping them to a lower dimension. Given a data matrix $X \in \mathbb{R}^{n \times p}$, the transformation matrix W can be obtained by solving the following optimization function:

$$\arg \min_{Y, W} \frac{1}{2} \|X - YW^T\|_F^2 \quad (2)$$

where $W \in \mathbb{R}^{p \times q}$, $W^T W = I_q$ and $Y \in \mathbb{R}^{n \times q}$. According to the Eckart-Young Theorem [7], the optimal solution is obtained when W consists of the q largest eigenvalues of $X^T X$. Therefore, the low dimensional mappings can be computed as follows:

$$Y = XW \quad (3)$$

Accordingly, we can approximately reconstruct the high dimensional X from the transformed data Y according via:

$$\hat{X} = YW^T \quad (4)$$

which represents the process of reconstructing high dimensional approximation using only low dimensional mappings and a transform matrix. Note that this process has low computational cost—we only need to calculate the inverse transformation of Y . In Section VI, we show that the reconstructed data is quite similar to original data. Therefore, using PCA alone for a data transformation does not satisfy the non-invertible criteria we introduced in Section IV, and thus insufficient for crafting an adversary-resilient DNN.

To deal with this problem and yet preserve computational efficiency, we equip PCA with our first non-invertible characteristic. To do this, we propose a novel dimension reduction method we call a designed linear mapping (DLM). DLM is designed to combine the lower dimensional mapping generated by PCA with other lower dimensional mappings generated by multiplying the original data with a column-wise, highly correlated transformation matrix. This design ensures that the PCA operation continues to preserve the critical information while the column-wise highly correlated transformation matrix guarantees that inverting the DLM will generate significant

reconstruction error. To explain the consequence of this, we now introduce DLM in detail and examine its properties.

Much as in (3), we shall formally define DLM to be:

$$Y = XC^T + \omega, \quad (5)$$

where $X \in \mathbb{R}^{n \times p}$ the same as introduced earlier, except that $Y \in \mathbb{R}^{n \times p_c}$. $\omega \in \mathbb{R}^{n \times p_c}$ denotes a normally distributed noise matrix, where each entry of ω generated from a normal distribution $N(0, \sigma^2)$. $C \in \mathbb{R}^{p_c \times p}$ is the transformation matrix obtained by following equation:

$$C = [B; A], \quad (6)$$

where C is constructed by combining a loading matrix $B \in \mathbb{R}^{p_b \times p}$ obtained via PCA with a designed matrix $A \in \mathbb{R}^{(p_c - p_b) \times p}$, of which all columns are highly correlated. This combination integrates PCA's information-preserving effects into our DLM. As such, the lower dimensional projection Y can provide a better representation of the original X .

Since the DLM described by (5) has a simple linear form, we estimate reconstruction \hat{X} for X using high-dimensional linear regression [28] (we omit calculation details due to space constraints). According to Theorem 1 in [28], under certain assumption, we can obtain a lower bound of the reconstruction error, which is the L_2 norm of the difference between X and \hat{X} as shown in (7):

$$\left(L_2(X, \hat{X})\right)^2 \geq \kappa_0 \sigma^2 \frac{s \log(p/s)}{p_c}, \quad (7)$$

where s denotes the sparsity of X . κ_0 is a constant whose value depends closely on the data set. Therefore, given a certain set of data, any linear transformation method is restricted by a constant lower bound calculated according to (7). In addition, according to Theorem 2 in [28], there also exists an upper bound of the reconstruction error as follows:

$$\left(L_2(X, \hat{X})\right)^2 \leq f(C) \frac{s \log(p)}{p_c}, \quad (8)$$

where $f(C)$ is a function of C . According to [28], the upper bound of the reconstruction error depends on both the data transformation matrix C and noise ω . When C is an independent correlation matrix, as in PCA, then the upper bound will approach the aforementioned lower bound hence restricting the reconstruction error within a narrow range close to the lower bound. However, since we specifically design C to be highly correlated, the upper bound will be significantly larger than the lower bound [9], [6], and thus result in a larger range for the reconstruction error.

B. Dimensionality Reduction by Learning an Invariant Mapping (DrLIM) DNN

As introduced in Section II, adversarial samples are generated by changing legitimate samples with small perturbations. But when processed by normal DNN models, the decisions made in a lower dimensional space are completely different from those made for legitimate samples, even though adversarial samples are highly similar to legitimate ones. Note that this characteristic also occurs in cross-model adversarial samples. Therefore, we intend to employ a dimensionality reduction method that preserves the similarity of high dimensional

samples in their lower dimensional mappings. Furthermore, our method needs to be capable of extracting critical information contained in the original data. Since the training of a DNN is already computationally intensive, our approach needs to be incremental in order to avoid the need for retraining the DNN.

Because of these considerations, we employ the dimensionality reduction method *DrLIM* proposed in [12]. DrLIM is specifically designed for preserving similarity between pairs of high dimensional samples when they are mapped to a lower dimensional space. The method restricts the lower dimensional mapping of cross-model adversarial samples to a vicinity where it is filled by mappings of legitimate samples that are highly similar to these cross-model adversarial samples. As a result, there is a significantly lower chance that an adversarial sample acts as an outlier in the lower dimensional space, since its mapped location is bounded by the mapped locations of similar, legitimate samples. DrLIM can also be used in an online setting.

More importantly, we theoretically prove that inverting DrLIM is an NP-hard problem. Therefore, DrLIM is suitable for our framework in that it satisfies the second characteristic of non-invertibility defined in Section IV. But first, we briefly review DrLIM (for a detailed explication please see [12]).

DrLIM consists of a convolutional neural network (CNN) model designed for optimizing the cost function:

$$\sum_{i=1}^P L(W, (Y, X_{i_1}, X_{i_2})^i), \quad (9)$$

where W denotes the coefficients. X_{i_1} and X_{i_2} denote the i th pair of input sample vectors with $i = 1 \dots P$. Y is a binary label assigned to each pair of samples, with $Y = 0$ denoting a similar pair of X_{i_1} and X_{i_2} , and $Y = 1$ for dissimilar pairs. Note that the similarity of each pair is not limited to any particular distance-based measure. This means that any prior knowledge can be utilized in representing similarity, including manually assigned similarity and dissimilarity. Thus, the classical approach for measuring a Euclidean distance between samples for representing dissimilarity can be enhanced with prior knowledge. Let the loss function for measuring the cost for each pair of samples be defined as:

$$L(W, Y, X_1, X_2) = (1 - Y) \frac{1}{2} (D(X_1, X_2))^2 + \frac{Y}{2} \{ \max(0, m - D(X_1, X_2)) \}^2, \quad (10)$$

where $D(X_1, X_2) = \|G(X_1) - G(X_2)\|_2$ is the Euclidean distance measured between the output lower dimension mapping $G(X_1)$ and $G(X_2)$ for the sample pair X_1 and X_2 . Let m be a predefined constant which indicates whether all dissimilar pairs are pushed or pulled towards to maintain a constant distance m (i omitted for simplicity).

Since G represents a mapping by the CNN to enable the recovery of high dimensional data from the low dimensional data $G(X)$, we need to first get $G^{-1}(X)$. For the forward pass of a conventional neural network, it is not guaranteed that the weight matrices are invertible [39], implying that information lost during pooling cannot be recovered. Thus, it is very difficult to compute $G^{-1}(X)$ and recover the original data from a low dimensional representation. Since inverting

the CNN is nearly impossible, one option is to reconstruct original X according to (10) given W and Y . In the following, we demonstrate that even this approach can be mapped to a NP-hard problem.

As discussed before, the most important property of DrLIM that allows it to fit into our framework is that it is provably non-invertible. Assuming $G(X)$ takes a simple linear form of $G(X) = WX$, then we have $D(X_1, X_2)^2 = (X_1 - X_2)^T W^T W (X_1 - X_2)$. Here we denote $\delta X = (X_1 - X_2)$. Following this assumption, we can reformulate (10) as follows:

$$\begin{aligned} \min_{\delta X, z} \quad & \sum (1 - Y) \delta X^T W^T W \delta X + Y z^2, \\ \text{s.t.} \quad & z \geq 0, \\ & z \geq m - \sqrt{\delta X^T W^T W \delta X}, \end{aligned} \quad (11)$$

where $z = \max(0, m - D(X_1, X_2))$. Here we reformulate the second constraint as $\sqrt{\delta X^T W^T W \delta X} \geq m - z$. Since $m - z \geq 0$, we have following:

$$\delta X^T W^T W \delta X \geq (m - z)^2. \quad (12)$$

Therefore, W is positive semi-definite. When both sides of (12) are multiplied by -1 and substituted into (11), we find that:

$$\begin{aligned} \min_{\delta X, z} \quad & \sum (1 - Y) \delta X^T W^T W \delta X + Y z^2, \\ \text{s.t.} \quad & z \geq 0, \\ & -\delta X^T W^T W \delta X \leq -(m - z)^2. \end{aligned} \quad (13)$$

From earlier work [36], the formulation (13) implies a quadratic problem with a non-positive semi-definite constraint, which is an NP-hard problem.

Note that solving (13) can yields the distance δX . There are multiple pairs of X_1 and X_2 that satisfy that $\delta X = (X_1 - X_2)$. This makes the problem even harder to solve. Additionally, since the linear relaxation (13) is already NP-hard, the original problem (10) is also NP-hard given that $G(X)$ is commonly regarded as a nonlinear function approximated by a neural network.

VI. EVALUATION

A. Experiments Settings

We evaluate our framework by performing experiments on the the widely-used MNIST data set [18]. MNIST contains a training split with 600000 greyscale images of handwritten digits and a test set containing 10000 images. Each image has a dimensionality of $28 \times 28 = 784$ pixels.

In the following experiments, we evaluate the proposed approaches under two types of adversarial samples. In order to first demonstrate that our mechanisms do indeed preserve the classification performance of the DNN, we test them with the original test set. We then test our methods with cross model adversarial samples to show that we achieve our secondary design goal.

B. Limitations of Adversarial Training

Recently, adversarial training has been shown to be effective in decreasing the classification error rates at test time [10], [24]. Adversarial training samples are generated using the fast gradient sign method (Section II). At test time, new ‘adversarial test samples’ are generated based on unseen, legitimate test samples, given that the original DNN model is readily accessible. In this case, adversarial training results in improved robustness.

We next build two different DNNs (model A and B) that share the same purpose—image recognition. Furthermore, we utilize adversarial training in learning both models A and B , which we denote as models A_{ADT} and B_{ADT} . Note that all following experiment results are the result of evaluating model A_{ADT} using different testing samples. The results appear in Table I. The second row ‘Legitimate’ presents the classification error rates achieved by model A_{ADT} when testing with normal samples. In the next third and fourth row, we show classification error rates using adversarial samples generated from model A and model B respectively. While the classification error in both settings is lower than 30%, the error rate obtained when testing with adversarial samples generated from model A itself is higher than the error rate found when testing with adversarial sample generated from a different model B . We speculate that this is possible because adversarial samples generated from a specific model might be more powerful for attacking that specific model. According to these results, adversarial training is indeed effective, depending on which external normal DNN model is used as the adversary.

However, as previously mentioned in Section III, DNN models learned using adversarial training are still vulnerable to unseen adversarial samples crafted specifically to target them. To demonstrate this limitation, we show the results of testing model A_{ADT} with new adversarial test samples generated from itself in the fifth row of Table I. The classification error rate is 78.10%, which is considerably higher than 25.06% when testing model A_{ADT} with adversarial testing samples generated from model A . This result is consistent with our previous analysis that the effectiveness of adversarial training is limited. In addition, we also present several visual samples of legitimate MNIST images as well as adversarial images generated from both model A and A_{ADT} in Fig. 4. As shown in Fig. 4, the legibility of image samples decreases as the attack power of these samples is increased. This indicates that an adversary would need to balance a trade-off between increasing attack strength and maintaining recognizable.

In the last row of Table I, we further show that the classification error rate of 57.15% when testing model A_{ADT} with adversarial samples generated from model B_{ADT} . Although the error rate is lower than 78.10%, it is still considerably

Table I. CLASSIFICATION PERFORMANCE OF TESTING AN ADVERSARIAL TRAINING ENHANCED MODEL WITH VARIOUS ADVERSARIAL SAMPLES

Different testing sets	Classification error rates of model A_{ADT}
Legitimate	0.0213
Adversarial testing samples from A	0.2506
Adversarial testing samples from B	0.1633
Adversarial testing samples from A_{ADT}	0.7810
Adversarial testing samples from B_{ADT}	0.5715



Figure 4. Examples of different legitimate adversarial samples from MNIST. The top row shows examples of legitimate images, in comparison with the middle and bottom rows, in which adversarial samples generated from model A and A_{ADT} are shown, respectively. Clear distortion of images can be observed in the bottom two rows. Moreover, the distortion is increased for adversarial samples generated from A_{ADT} . This implies that model A_{ADT} is more robust than model A , which can be attributed to adversarial training. More specifically, an attack on model A_{ADT} requires yet further perturbation which makes it easier for humans to identify the assault.

higher than both 16.33% and 25.06% shown in the third and fourth rows. This result demonstrates that adversarial samples generated from enhanced DNN models maintain their cross-model efficacy. It is important to note that the effectiveness of the attack varies across models, an effect observed when using ordinary adversarial samples.

C. Classification Performance

1) *Classification Performance of DLM-DNN*: In this experiment, we fix the reduced dimensionality to 100. These mappings are found by DLM and PCA. In order to better explore the effect of combining DLM with PCA, we vary the percentage P_{pca} of PCA mappings used in the fixed 100 dimension. Meanwhile, the percentage of DLM mappings used varies according to $100 - P_{pca}$. In addition, we also change the level of noise added to study its influence on classification performance.

We first show the classification performance when testing with legitimate samples in the column named ‘*Legitimate*’ in Table II. The noise coefficient is set to be either 0.1 or 0.3, while P_{pca} varies from 5% to 95%. We first notice that our proposed DLM-DNN does result in any significant performance degradation compared to adversarial training, especially when the noise coefficient is equal to 0.1. However, it is noticeable that with a larger noise coefficient of 0.3, the classification error rate of DLM-DNN goes up. This performance degradation is due to the increase of noise injected into the lower dimensional mappings. Therefore, we conclude that if properly set, DLM-DNN can result in performance comparable to adversarial training.

We further examine the influence of varying P_{pca} on classification performance. As shown in Table II, the classification performance slightly improves in this case. This also happens when the noise coefficient is set to 0.3. To better explain the performance boost, we further show the variance in preservation rates as a function of different dimensionalities using PCA in Fig. 5. We show that PCA preserves over 90% of the critical information within samples from the MNIST data set when mapping original data to the 100 dimensional space. However, according to Fig. 5, when $P_{pca} = 5\%$, information preserved by PCA is insufficient for representing the original samples well. Meanwhile, the 95% mapping obtained from DLM is not as effective as PCA mappings for representing the original samples. Therefore, this combination leads to the highest classification error rates for both noise coefficient

Table II. CLASSIFICATION PERFORMANCE OF DLM-DNN AND DR LIM-DNN

Trained Model		Classification error rates with different testing sets	
		Legitimate	Adversarial
Normal DNN		0.0198	0.8981
Adversarial Training Enhanced DNN		0.0213	0.2506
DLM-DNN	Noise coefficient of 0.1	PCA(95%)	0.0226
		PCA(75%)	0.0247
		PCA(50%)	0.0258
		PCA(25%)	0.0268
		PCA(5%)	0.3101
	Noise coefficient of 0.3	PCA(95%)	0.0386
		PCA(75%)	0.0403
		PCA(50%)	0.0427
		PCA(25%)	0.0452
		PCA(5%)	0.3710
DrLIM-DNN		0.0384	0.1380

equal to 0.1 and 0.3. However, as P_{pca} reaches 25%, enough information is preserved resulting in only a slight decrease in classification error. Meanwhile, when P_{pca} varies from 25% to 95%, the benefit of preserving any further information diminishes as with only a negligible decrease in the error rate.

We next evaluate the classification performance of DLM-DNN when confronted with adversarial samples. We list the classification error rates in the column noted as ‘*Adversarial*’. According to Table II, the error rates obtained by the DLM-DNN are considerably lower than that of a standard DNN, 0.8981. Again, when P_{pca} is properly set, the DLM-DNN achieves results comparable to adversarial training. The highest error rate is obtained when $P_{pca} = 5\%$, for either noise coefficient is set to 0.1 or 0.3. This is consistent from our previous analysis for testing with legitimate samples. Interesting enough, as P_{pca} ranges from 25% to 95%, classification error goes up. This observation might imply that the impact of adversarial samples is mitigated to a larger degree when more

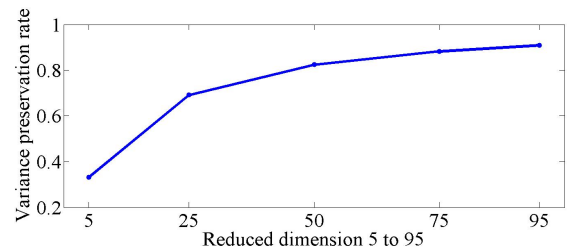


Figure 5. Variance preservation rates with different reduced dimensions of PCA.

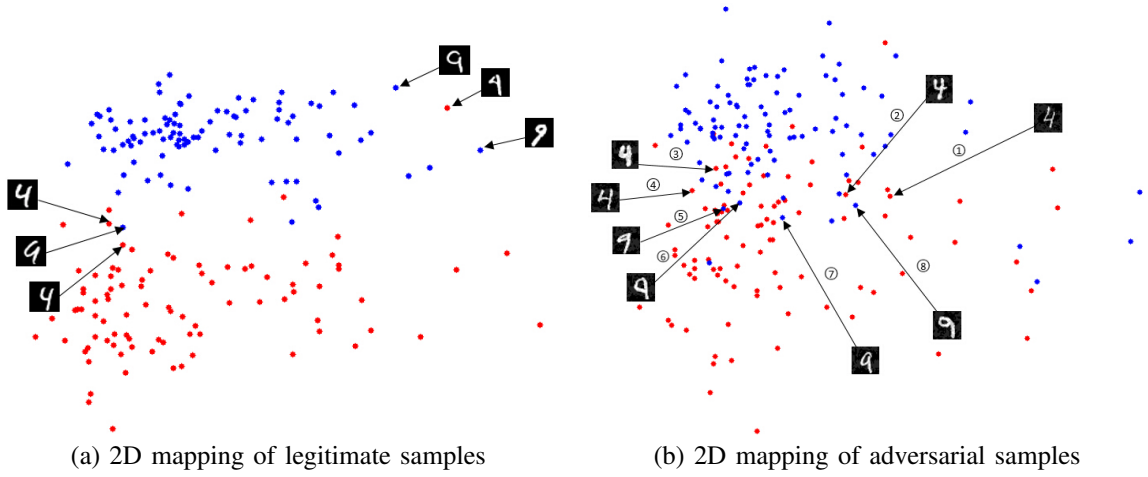


Figure 6. 2D mapping generated by DrLIM. In both (a) and (b), we show the 2D projection of 200 legitimate samples and 200 corresponding adversarial samples. Among these samples, half are randomly selected from class 4 (red dots) while the other half is randomly selected from class 9 (red dots). In (a), we highlight two outliers from both class 4 and 9 as well as their neighbours. Similarly, we highlight 8 outliers from both classes and mark each outlier sequentially by assigning each with an index, ranging from 1 to 8.

random disturbances are added.

2) DrLIM-DNN Classification Performance: In this experiment we demonstrate the classification performance of DrLIM-DNN. The training set used for evaluation includes 5 classes from the MNIST data, and each class contains 2000 samples. For testing, each of the 5 classes contains 1000 testing samples. The adversarial testing samples are generated based on this testing set.

For training the DrLIM, we label a pair of image samples as similar when they have the same label. This simplifies the training of DrLIM utilizing strong prior knowledge. We further set the reduced dimension to 30 during the experiments. Classification performance is shown in the bottom part of Table II. According to these results, using DrLIM-DNN results in a slightly higher error rate (0.0384) when testing with legitimate samples, but achieves a significant improvement in performance (0.1380) when testing adversarial samples. Especially in the latter case, DrLIM-DNN shows higher robustness when compared to adversarial training.

As previously introduced in Section V, DrLIM is designed with the objective of preserving similarity between a pair of high dimensional samples when mapped to lower dimensional space. In order to better illustrate this property, we first map 200 legitimate examples of MNIST image samples into 2D dimension using DrLIM. According to the visualization of the 2D mapping shown in Fig. 6(a), DrLIM effectively preserves the similarity of legitimate samples in the mapped 2D space. We notice some outliers and hence highlight them and their neighbours by showing their corresponding images. For example, in Fig. 6(a), one of the outliers is a blue dot representing 9 surrounded by two red dots representing 4. Meanwhile the other outlier is a red 4 neighbouring to two blue 9s. As shown in Fig. 6(a), these outliers are hard to recognize, even for human.

Since the point of DrLIM is to preserve the similarity in a lower dimensional space, we further visualize the 2D mapping of 200 adversarial samples in Fig. 6(b). Note that these adversarial samples are generated from those 200 legitimate

samples selected before. The 2D mapping in this case is not as clear as that for legitimate samples, especially since the overlapping is more obvious. However, the similarity between pairs of samples are still reasonably well-preserved. In order to explore more of these outliers, in Table III, we show the probabilities of making wrong classification decisions when testing a normal DNN and a DrLIM-DNN with these outliers. As shown in Table III, these outliers cause a normal DNN to make wrong classification results with over 97% confidence. However, when processed with DrLIM-DNN, although these outliers are not mapped to ideal regions, the probabilities of being wrongly classified is significantly reduced to lower than 66%. This result indicates that a DrLIM-DNN is effective for responding to unfamiliar samples with lower confidence. Therefore, DrLIM-DNN will not suffer as much as a normal DNN would when confronted with highly confusing adversarial samples.

As our experimental results show, DrLIM-DNN provides the best performance when tested against adversarial samples. In particular, the DrLIM-DNN achieves a classification error rate 44.93% (smaller than the classification error rate obtained using adversarial training). In addition, DrLIM-DNN also shows its effectiveness for avoiding making strongly confident predictions for adversarial samples. With respect to this property, there has been limited research that shows similar results. However, since the DrLIM requires also building a CNN model, the computational cost is much higher compared

Table III. CLASSIFICATION CONFIDENCE OBTAINED FROM NORMAL DNN AND DRLIM-DNN

Outlier No.	Classification confidence of testing adversarial samples	
	Normal DNN	DrLIM-DNN
1	0.9995	0.5196
2	0.9721	0.5290
3	0.9989	0.6220
4	0.9921	0.5646
5	0.9998	0.5903
6	0.9997	0.5402
7	0.9998	0.6596
8	0.9919	0.5638

to both DLM-DNN and adversarial training. Therefore, to best utilize the advantages of DrLIM-DNN, it might be reasonable to use a smaller data set with a fewer number of classes.

D. Reconstruction Performance

As previously introduced in Section V, both DLM-DNN and DrLIM-DNN are non-invertible for different reasons. More importantly, we have proven that recovering the original data from a low dimensional space induced by DrLIM is an NP-hard problem. In this subsection, we mainly focus on inverting the proposed dimensional reduction method DLM by approximating it with a linear transformation matrix. We obtain the linear transformation matrix by solving a linear regression problem. In case the original data is sparse, we further employ a linear regression with L_1 regularization. First, we demonstrate that when configuring DLM as pure PCA, the approach is not robust given that it may be effectively inverted and thus allow for reconstruction of adversarial samples. Next, we examine the reconstruction error obtained from inverting DLM, taking a percentage of PCA mappings less than 100%.

We evaluate the reconstruction performance when inverting one extreme case of DLM, where DLM uses only PCA mappings. In this way, the original samples are only processed by PCA before being input to a DNN model. We refer to this method as PCA-DNN for comparison. Therefore, an adversary can easily recover the adversarial images given their lower dimensional mappings. To examine this extreme case, we first configure DLM as pure PCA and map legitimate testing samples to a 100-dimensional space. Then we reconstruct these legitimate samples by inverting PCA, as explained in Section V. To demonstrate the effectiveness of inverting PCA, we show reconstructed legitimate samples in Fig. 7(a). In this case, the reconstruction was successful and the samples are hardly different from legitimate ones. Now we assume that an adversary has acquired the lower dimensional mappings generated by PCA. According to Section II, this adversary can easily generate their corresponding *lower dimensional adversarial mappings*. Following the inversion procedure of Section V, this adversary can further reconstruct adversarial samples based on these lower dimensional adversarial mappings. We also show the reconstructed adversarial samples in Fig. 7(b). While

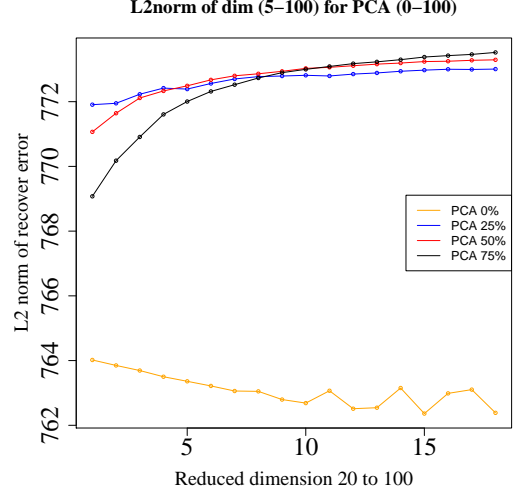


Figure 8. Reconstruction errors with varying parameters for inverting DLM. For each curve, it represents the reconstruction errors generated by setting reduced dimensions in the range of (20, 100). Meanwhile, the percentage of PCA mapping is fixed to any value of 5%, 25%, 50%, 75% and 95%.

Fig. 7(a) and Fig. 7(b) are difficult to differentiate, we use the reconstructed adversarial samples to test a normal DNN model and a DLM-DNN under different settings. According to the testing results shown in Table IV, the reconstructed adversarial samples maintain their attack power against a normal DNN model. The classification error rate in this case is larger than 60%. In contrast, these adversarial samples can be effectively defended by a DLM-DNN as shown in Table II.

We finally investigate the reconstruction errors when inverting DLM-DNN. Note that since we have shown that using PCA is not secure for defending against reconstructed adversarial samples, the reconstruction errors for this extreme case will not included in following discussion. We present reconstruction errors when varying percentages of PCA mappings used and when varying sub-space dimensionality in Fig 8. Our experiment shows that inverting a DLM-DNN leads to high reconstruction errors, regardless of how many PCA mappings are used what dimensionality is used. Recall the theoretical analysis of DrLIM-DLM in Section V, we demonstrate that our proposed methods effectively build an adversary-resistant DNN.

Table IV. CLASSIFICATION PERFORMANCE OF PCA-DNN AND DLM-DNN TESTING WITH RECONSTRUCTED ADVERSARIAL SAMPLES BY INVERTING PCA

Trained models		Classification error rates of testing with reconstructed adversarial samples
Normal DNN		0.6596
Noise Coefficient of 0.1	PCA(95%)	0.2846
	PCA(75%)	0.2011
	PCA(50%)	0.1447
	PCA(25%)	0.1131
	PCA(5%)	0.3691
Noise Coefficient of 0.3	PCA(95%)	0.1864
	PCA(75%)	0.1729
	PCA(50%)	0.1449
	PCA(25%)	0.1884
	PCA(5%)	0.4766

VII. RELATED WORK

In this work, we build adversary-resilient DNN architectures using non-invertible dimension reduction methods. Related research commonly falls into the area of adversarial machine learning or dimensionality reduction methods. In this section we introduce several state-of-the-art adversarial machine learning technologies. These technologies can be categorized as either data augmentation or DNN model-complexity enhancement. Finally, we present several prevalent dimensionality reduction methods and study their suitability under our proposed framework.

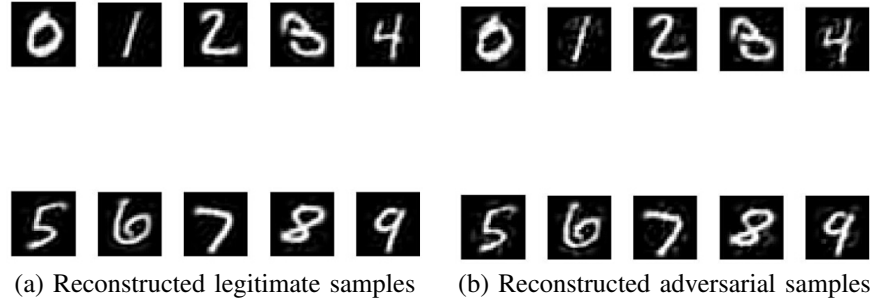


Figure 7. Reconstructed samples by inverting PCA-DNN. The reconstructed legitimate and adversarial samples are shown in (a) and (b) respectively. Specifically, the former are reconstructed based on the lower dimensional mappings of legitimate samples. As for the latter, reconstruction is conducted based on lower dimensional adversarial mappings.

A. Adversarial Machine Learning

Approaches to the robustness issue mainly focus on adversarial training and enhancing DNN complexity.²

Adversarial training approaches usually augment the training set by integrating adversarial samples [35]. More formally, this data augmentation approach can be theoretically interpreted regularizing the training objective function. This regularization is designed to penalize certain gradients using the derivative of the loss with respect to the input variables. The penalized gradients indicate a direction in the input space that the objective function is most sensitive to. For example, recent work [10] introduced such an objective function, directly combining a regularization term with the original cost function. Instead of training a DNN with a mixture of adversarial and legitimate samples, other approaches [23] use only adversarial samples. Despite the difference between these strategies, a unifying framework, *DataGrad* [24], was proposed to generalize adversarial training of deep architectures and help explain prior approaches [11], [29].

However, since adversarial training still falls into the category of data augmentation, which, as explained earlier in this paper, cannot possibly hope to cover the entire infinite space of possible adversarial samples. In other words, adversarial training is still vulnerable to unforeseen adversarial samples. Therefore, we looked to improving the robustness of DNNs by enhancing the model complexity.

Recent work [26], denoted as *distillation* enhances a normal DNN by enabling it to utilize other informative inputs. More specifically, for each input training sample, instead of using its original *hard label* which only indicates whether it belongs to a particular class, the enhanced DNN utilizes a *soft label*. This means that probability distribution over all classes is used as a corresponding target vector. In order to use this more informative soft label, two DNNs must be designed successively and then stacked together. One DNN is designed for generating more informative input samples while the other DNN is trained using these more informative samples in the usual way. However, as discussed in section III, once the architecture of distillation is disclosed to attackers, it is always possible to find adversarial samples based on this very defense mechanism. If an attacker [3] generates adversarial samples by artificially reducing the magnitude of the input to the final

softmax function and modifies the objective function used for generating adversarial samples [25], the defensively distilled networks are open to assault. A denoising autoencoder [11] approach was also proposed to filter out adversarial perturbations. However, it too suffers from the same problem as [26]. Again, once the architecture of the stacked DNN and auto-encoder is disclosed, attackers can still generate adversarial samples by treating this new model as a normal feed-forward DNN only with more hidden layers. Indeed, adversarial samples of this stacked DNN [11] show even smaller distortion compared with adversarial samples generated from the original DNN. Thus, a successful method for increasing model complexity requires the property that even if the model architecture is disclosed, it would not be possible to generate adversarial samples based on this information.

B. Dimensionality Reduction Methods

From the previous analysis in Section IV, we see that dimensionality reduction has definite advantages for the design goals of our proposed architecture using non-invertible data transformations. As such, we investigated various dimensional reduction methods including both linear and nonlinear methods.

Recall from Section V, dimensionality reduction methods that preserve the similarity between high dimensional samples in the mapped lower dimensional space can be helpful in defending against adversarial samples. This is due to the fact that adversarial samples are highly similar to legitimate samples yet can be wrongly classified in a lower dimensional space. One popular linear dimension reduction method with aforementioned characteristic is multi-dimensional scaling (MDS) [17]. MDS is computationally efficient and has been widely adopted in various machine learning tasks. More importantly, MDS generates lower dimensional mappings based on similarity, which is usually represented by Euclidean distance, between high dimensional data points. Given this property, an adversarial sample will not be mapped that far away from its similar, legitimate neighbors. However, MDS have three shortcomings which restrict its usage in our proposed framework. First, since MDS is built only with distance information between samples, critical feature information contained in data samples cannot be preserved. Second, MDS cannot provide a mapping from high dimensional input to a lower dimensional output when being applied to unseen data samples. When stacking DNN on top of MDS, whenever unseen samples appears, the entire combined

²For details of robustness issues, please see Section II.

architecture needs to be retrained, which is computationally undesirable.

We also investigated another nonlinear dimensional reduction methods of t-SNE [19]. Specially, t-SNE is similar to MDS in that it is also capable of preserving the similarity between high dimensional data in a lower dimensional space. t-SNE minimizes KL-divergence between two probability distributions, one representing the probability of being neighbors in a high dimensional space while the other represents the probability of being neighbors in a lower dimensional space. However, similar to MDS, t-SNE is not suitable for online processing.

The dimensionality reduction methods used in our proposed framework take these aforementioned issues into account. In particular, DLM is more computationally efficient in that it can easily preserve feature information of the original data samples. Meanwhile, DrLIM not only preserves similarity and important feature information of original input space simultaneously, but is an incremental algorithm. More importantly, both DLM and DrLIM are non-invertible which we justify in theoretically and empirically in Sections V and VI.

VIII. CONCLUSION

We proposed a new framework for constructing deep neural network models that are robust to adversarial samples. Our framework design is based on an analysis of both the “blind-spot” of DNNs and the limitations of currently proposed solutions.

Using our proposed framework, we developed two adversary-resilient DNN architectures that leverage non-invertible data transformation mechanisms. Using the first proposed approach for processing the data fed into DNN models, we empirically showed that crafting an adversarial sample for this architecture will incur significant distortion and thus lead to easily detectable adversarial samples. In contrast, under the second architecture, we theoretically demonstrated that it is impossible for an adversary to craft an adversarial sample to attack it. This implies that our proposed framework no longer suffers from attacks that rely on generating model-specific adversarial samples.

Furthermore, we demonstrated that recently studied adversarial training methods are not sufficient defense mechanisms. Applying our new framework to the MNIST data set, we empirically demonstrate that our new framework significantly reduces the error rates in classifying adversarial samples. Furthermore, our new framework has the same classification performance for legitimate samples with negligible degradation. Future work will entail investigating the performance of our framework in a wider variety of applications.

REFERENCES

- [1] Y. Bar, I. Diamant, L. Wolf, and H. Greenspan. Deep learning with non-medical training used for chest pathology identification. In *SPIE Medical Imaging*, pages 94140V–94140V. International Society for Optics and Photonics, 2015.
- [2] I. G. Y. Bengio and A. Courville. Deep learning. Book in preparation for MIT Press, 2016.
- [3] N. Carlini and D. Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016.
- [4] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu. Large-scale malware classification using random projections and neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3422–3426. IEEE, 2013.
- [5] A. d’Aspremont and S. Boyd. Relaxations and randomized methods for nonconvex qcqps. *EE392o Class Notes, Stanford University*, 2003.
- [6] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, April 2006.
- [7] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.
- [8] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Scene parsing with multiscale feature learning, purity trees, and optimal covers. *arXiv preprint arXiv:1202.2160*, 2012.
- [9] M. Fornasier and H. Rauhut. Compressive sensing. In *Handbook of mathematical methods in imaging*, pages 187–228. Springer, 2011.
- [10] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *CoRR*, 2014.
- [11] S. Gu and L. Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv:1412.5068 [cs]*, 2014.
- [12] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, volume 2, pages 1735–1742, 2006.
- [13] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.
- [14] G. E. Hinton. Learning multiple layers of representation. *Trends in cognitive sciences*, 11(10):428–434, 2007.
- [15] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári. Learning with a strong adversary. *CoRR, abs/1511.03034*, 2015.
- [16] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.
- [17] J. B. Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129, 1964.
- [18] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998.
- [19] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.
- [20] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing with virtual adversarial training. *stat*, 1050:25, 2015.
- [21] J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, and A. Y. Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 265–272, 2011.
- [22] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436. IEEE, 2015.
- [23] A. Nøklund. Improving back-propagation by adding an adversarial gradient. *arXiv:1510.04189 [cs]*, 2015.
- [24] A. G. Ororbis II, C. L. Giles, and D. Kifer. Unifying adversarial training algorithms with flexible deep data gradient regularization. *arXiv:1601.07213 [cs]*, 2016.
- [25] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016.
- [26] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1511.04508*, 2015.
- [27] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas. Malware classification with recurrent networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1916–1920. IEEE, 2015.
- [28] G. Raskutti, M. J. Wainwright, and B. Yu. Minimax rates of estimation for high-dimensional linear regression over-balls. *IEEE Transactions on Information Theory*, 57(10):6976–6994, 2011.

- [29] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 833–840, 2011.
- [30] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.
- [31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.
- [32] E. C. R. Shin, D. Song, and R. Moazzezi. Recognizing functions in binaries with neural networks. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 611–626, 2015.
- [33] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- [34] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [35] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.
- [36] S. A. Vavasis. *Nonlinear Optimization: Complexity Issues*. Oxford University Press, Inc., New York, NY, USA, 1991.
- [37] Y. Xu, T. Mo, Q. Feng, P. Zhong, M. Lai, I. Eric, and C. Chang. Deep learning of feature representation with multiple instance learning for medical image analysis. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1626–1630. IEEE, 2014.
- [38] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue. Droid-sec: Deep learning in android malware detection. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 371–372. ACM, 2014.
- [39] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.